*Fritz-Haber-Institut der Max-Planck-Gesellschaft • Abt. Theorie*

# The FHIMD Toolkit

✧✧✧

# User's Manual

Andreia L. da Rosa, Seung Mi Lee & Evgeni Penev

version 1.0, April 2002

# Preface

This manual covers a set of software packages and routines that can be used to extract specific information from the output generated by the `fhi98md` computer code [1, 2].

Chapter 1 describes the `murn` package for performing fits with the Murnaghan equation of state. It consists of a FORTRAN77 source file murn.f (Sec. 1.2) and the Bourn-shell script `murfit` (Sec. 1.3).

Chapter 2 is devoted to the `Contour Plot` package, written in FORTRAN90. It provides functionalities for post-processing the standard output from the `fhi98md` code so as to allow you visualizing charge densities and simulating STM images. The package makes calls to standard FFT library functions (inspect the fft.f90 source file) so the proper library should be linked upon making. If you have got the `fhi98md` program installed, you may eventually use the libfft.a library archive which resides in the `lib/` subdirectory of the program's directory tree.

The description in Chapter 3 covers the subroutine `angular` added to the latest version of the `fhi98md` program code. The functionalities implemented in angular.f allow (run-time) calculation of the total, atom-, and state-resolved densities of states. The routine-specific input and output are described in Sec. 3.2.1 and Sec. 3.2.2, respectively.

Further information and the package downloads can be found on the `fhi98md` home page <http://www.fhi-berlin.mpg.de/th/fhi98md/>.

# Contents


**1 The Murnaghan fit** **1**
- 1.1 Getting started: Murnaghan equation of state . . . 1
- 1.2 Implementation: the `murn` package . . . 2
  - 1.2.1 Input . . . 3
  - 1.2.2 Output . . . 4
- 1.3 Command-line interface: `murfit` . . . 6
- 1.4 Example: equilibrium bulk lattice constant of GaAs in LDA . . . 8

**2 `Contour Plot` Package** **15**
- 2.1 Contour Plots . . . 15
  - 2.1.1 STM simulation . . . 15
  - 2.1.2 Total charge density and charge density difference . . . 16
  - 2.1.3 Single state . . . 16
- 2.2 Input file . . . 16
  - 2.2.1 Format . . . 16
  - 2.2.2 Defining the plane and the range of the contour plot . . . 17
  - 2.2.3 Setting the plotting options . . . 18
- 2.3 Output . . . 20
  - 2.3.1 Format . . . 20
  - 2.3.2 STM simulation, charge density, and single state . . . 21
  - 2.3.3 Charge density difference . . . 21
- 2.4 Shell script . . . 21
- 2.5 Examples . . . 24
  - 2.5.1 Example 1: total charge density for GaAs bulk . . . 24
  - 2.5.2 Example 2: charge density for a specific state of GaAs bulk 24
  - 2.5.3 Example 3: STM simulation for GaAs (110) . . . 25

**3 Projected Density of States** **27**
- 3.1 Some definitions . . . 27
- 3.2 Shell script . . . 27
  - 3.2.1 Input . . . 30
  - 3.2.2 Output . . . 31
  - 3.2.3 Format . . . 31
- 3.3 Examples . . . 32
  - 3.3.1 DOS and PDOS for GaAs bulk . . . 32

# Chapter 1

# The Murnaghan fit

## 1.1 Getting started: Murnaghan equation of state

In elasticity theory one can easily show that the free energy of deformation of an isotropic medium can be written in the form [3]

$$F = F_0 + \mu\left(\varepsilon_{ik} - \frac{1}{3}\delta_{ik}\mathrm{Tr}\varepsilon\right)^2 + \frac{1}{2}B\sum_{i=1}^{3}\varepsilon_{ii}^2, \tag{1.1}$$

where $\varepsilon_{ik}$ is the strain tensor, and $\mu$ and $B$ are respectively the shear and bulk modulus of the medium. The validity of this general second-order expansion of $F$ is restricted to small deformations $\varepsilon_{ik}$, and it is commonly implied that $F_0$ is the free energy of the undeformed medium, $\varepsilon_{ik} = 0$. Under these assumptions $\mu$ and $B$ are just elastic *constants*.

Murnaghan has put Eq. (1.1) in more general context by addressing the dependence of the elastic moduli on the *initial* value of stress [4]. His underlying supposition is that, as a first approximation, the bulk modulus $B$ of a medium under uniform pressure $P$ is linear in $P$,

$$B = B(T,V) = -V\left(\frac{\partial P}{\partial V}\right)_T := c(1+kP), \tag{1.2}$$

where $c$ and $k$ are coefficients in the first-order polynomial. Simple integration of Eq. (1.2) leads to the equation

$$\frac{V_0}{V} = (1+kP)^{1/ck}, \tag{1.3}$$

originally proposed by Murnaghan in Ref. [4]. Note that $V_0$ is the equilibrium (initial) volume ($V_0 = V(P=0)$) with respect to which changes of $V$ are measured. Setting $\Delta V = V_0 - V$ we rewrite Eq. (1.3) in the form

$$\frac{\Delta V}{V_0} = 1 - (1+kP)^{-1/ck}, \tag{1.4}$$

an expression which turns out to reasonably fit the experimental data for pressures even up to $10^5$ atm. (cf. Ref. [4]).

For the purposes of computing equilibrium properties of bulk materials it is desirable to make a step further and find corresponding expression for the free energy $F$. For the ease of derivation let us assume that the deformations are isothermal and thus the differential of $F$ is simply

$$dF = -PdV. \tag{1.5}$$

Eq. (1.3), on the other hand, can be used to obtain $P = P(V)$ and substitute it in (1.5). The integration of the latter gives

$$\begin{aligned} F(T,V) - F(T,V_0) &= \frac{1}{k}\int_{V_0}^{V}\left(\frac{V_0}{v}\right)^{ck} dv + \frac{1}{k}\int_{V_0}^{V} dv \\ &= \frac{V}{k(ck-1)}\left[\left(\frac{V_0}{V}\right)^{ck} + ck - 1\right] - \frac{cV_0}{ck-1}. \end{aligned} \tag{1.6}$$

The linearity of $B$ in $P$, as assumed in Eq. (1.2), implies that

$$c = -V\left(\frac{\partial P(T,V)}{\partial V}\right)_{T,P=0} \equiv B_0, \qquad ck = -\frac{\partial}{\partial P}\left(V\frac{\partial P(T,V)}{\partial V}\right)_{T,P=0} \equiv B_0'. \tag{1.7}$$

In terms of $B_0$ and $B_0'$ the Murnaghan equation of state, Eq. (1.6), takes the final form

$$F(V) - F(V_0) = \frac{B_0 V}{B_0'}\left[\frac{(V_0/V)^{B_0'}}{B_0' - 1} + 1\right] - \frac{B_0 V_0}{B_0' - 1}, \tag{1.8}$$

which is implemented in the `murn` program, described in the next section.

## 1.2 Implementation: the `murn` package

The gist of calculating the equilibrium lattice constant of bulk materials as implemented in the `murn` package lies in the evaluation of Eq. (1.8) for sufficient number of values of the lattice parameter $a$ and then performing a least-squares fit to the energy versus $a$ dependence. The main utility of the `murn` package is a FORTRAN77 program, source file `murn.f` (it is hard to trace who is the original author(s), but you could still check the comments therein). It evaluates Eq. (1.8) (subroutine `murng1`), performs least-squares fit to the calculated points and extracts the equilibrium lattice constant $a_0$, bulk modulus $B_0$, and its pressure derivative $B_0'$. Optionally the program can calculate corrections to the energy to account for zero-point lattice vibrations. If, for example, you have got the GNU-project FORTRAN77 compiler `g77` already installed on your computer making of an executable (say, called `murn.x`) can be done in a single line:

```
$ g77 -o murn.x murn.f
```

In the following we describe the required input, and the format of the output generated by the `murn` program. If you intend to use the `murfit` command-line interface which is part of the `murn` package, and documented in Sec. 1.3, it is necessary to get familiar with all input parameters (consult the "gray" table below) so that you will be able to master them via command line options.

### 1.2.1 Input

The `murn` program reads all input data from the standard input. Thus, if one has the input saved in the file file.inp and wants the output to be stored in the file file.out the following command line will do the right job for you:

```
$ murn.x < file.inp > file.out
```

The input file file.inp is ASCII file and must contain the following data records:

file.inp

```
 1 3                          : iunit  /1=Ry, 2=eV, 3=Ha/
 2 1.00                       : convav /bcc=0.5, fcc=0.25, pc=1.0/
 3 10.0 11.9 100              : alat_min, alat_max, nr_alat
 4 .false. 2.19 428.0 109.9   : tzero, gamma, thetad, atvol
 5 12                         : nnn    /number of input data lines/
 6 10.0 -71.90336991          : alat  eninp
 7 10.4 -71.93052914          : alat  eninp
 8 10.7 -71.94080817          : alat  eninp
   ⋮
17 11.9 -71.92865989          : alat  eninp
```

Format description:

| Field | Parameter | Kind | Description |
|---|---|---|---|
| 1 | *iunit* | Integer | Units in which energy is measured. Allowed values:<br>`1` - Rydberg<br>`2` - eV<br>`3` - Hartree |
| 2 | *convav* | Real | Conversion factor from $a^3$ to the volume of the unit cell $\Omega$ used to calculate the energy. For example, if zinc-blende structure is represented as simple cubic unit cell, $\Omega = a^3$, set this to `1.0`; the same structure can be represented with face-centered cubic unit cell, $\Omega = \frac{1}{4}a^3$, in which case the conversion factor has to be set to `0.25`; for hexagonal unit cells $\Omega = \left(\frac{\sqrt{3}}{2}\frac{c}{a}\right)a^3$. |
| 3.1 | *alat_min* | Real | Lower bound of the interval of change of the lattice parameter $a_{\min}$, $a \in [a_{\min}, a_{\max}]$ |
| 3.2 | *alat_max* | Real | Upper bound of the range of change of the lattice parameter $a_{\max}$. |
| 3.3 | *nr_alat* | Integer | Number of mesh points $n$ used to evaluate the fitting polynomial, $a_m = a_{\min} + (m-1)(a_{\max} - a_{\min})/(n-1)$. Larger values result in smoother curves for plotting. |

| Field | Parameter | Kind | Description |
|---|---|---|---|
| 4.1 | *tzero* | Logical | Flag to toggle on/off inclusion of correction for zero-point lattice vibrations á la Janak-Moruzzi-Williams [5], $$E_{\mathrm{vib}} = \frac{9}{8} k_{\mathrm{B}} \Theta_{\mathrm{D}}(\Omega), \quad \Theta_{\mathrm{D}}(\Omega) = \Theta_{\mathrm{D},0} \left( \frac{\Omega_0}{\Omega} \right)^{\gamma}.$$ This feature is tested only for fcc structure of Al. The following three fields take effect only if *tzero*=`.true.`. |
| 4.2 | *gamma* | Real | Grüneisen's constant, $\gamma$ |
| 4.3 | *thetad* | Real | Debye temperature at $\Omega_0$, $\Theta_{\mathrm{D},0}$ |
| 4.4 | *atvol* | Real | Volume per atom, $\Omega_0$ |
| 5 | *nnn* | Integer | Number of ($a$,energy) pairs to be read in, each entering as a separate field. Maximum allowed value is 50. |
| 6.1 | *alat* | Real | Lattice constant $a$. |
| 6.2 | *eninp* | Real | Calculated energy at $a$. |
| | $7 \ldots 5 + nnn$ | | *nnn* pairs (*alat*,*eninp*) must enter in fields 6,7,...,5+*nnn*. |

### 1.2.2 Output

The output from the `murn` program consists of two parts: `#`-commented header, and numeric table representing the Murnaghan fit to the input data according to Eq. (1.8).

Header The header part is, basically, a copy of the input to the `murn` program: volume of the unit cell, conversion factor, bounds of the sampled range of lattice constants, etc. The actual result from the fitting procedure goes in the section

```
#----------
# fit by murnaghan equation equation of state
#----------
```

As the least-squares fit is implemented on an iterative principle (minimization of the absolute deviation $|E_{\mathrm{fit}} - E_0|^2$ between the energy according to the Murnaghan equation of state $E_{\mathrm{fit}}$ and the calculated value $E_0$) the first field of the following record (`iter`) gives the number of iterations. The next four fields are of particular importance:

`alatt0` the equilibrium lattice constant, obtained from

the minimum of the Murnaghan fit:

$$a_0 \quad \leftarrow \quad \min_{a\in[a_{\min},a_{\max}]} E_{\rm fit}(a).$$

| | |
|---|---|
| `vol0` | the corresponding equilibrium volume |
| `b0` | bulk modulus $B_0$ in mbar. |
| `b0prime` | derivative of the bulk modulus $B_0'$. |

The last three fields give the fitted energy (`e0`) at the minimum, and some statistical information about the fit (`sq sum`, `ierr`). One extracts the *converged* $a_0$, $B_0$, and $B_0'$ from the last record (see the shading in file.out below). For a converged result `ierr` $= 0$.

Numeric table The length of the table is determined by the input parameter *nr_alat*. The sacramental Murnaghan fit consists of the first and second columns which give, respectively, the mesh point $a_m$, and the fitted energy value $E_{\rm fit}(a_m)$. The third column is just the fitted energy measured with respect to the minimum of the input energies, i.e. $E_{\rm fit} - \min_a E_0(a)$.

file.out

```
# unit of length =       0.529177  angstroem
# energies input in HARTREES
# unit cell volume = a**3 *       1.000000
# a(min)=          10.0000000
# a(max)=          11.9000000
# points: 100
# number of data points: n =   12
#----------
#    i     alatt         volume              E
#     1      10.000000    1000.000000     -71.9033699100
#     2      10.400000    1124.864000     -71.9305291400
#     3      10.700000    1225.043000     -71.9408081700
#     4      10.800000    1259.712000     -71.9430085400
#     5      10.900000    1295.029000     -71.9433859700
#     6      11.000000    1331.000000     -71.9443174400
#     7      11.100000    1367.631000     -71.9431787800
#     8      11.200000    1404.928000     -71.9434457300
#     9      11.300000    1442.897000     -71.9420208700
#    10      11.400000    1481.544000     -71.9411302200
#    11      11.600000    1560.896000     -71.9375102700
#    12      11.900000    1685.159000     -71.9286598900
#----------
#     alatt0   vol0 (ula**3)  b0 (mbar)     b0prime        e0
#   11.0000   1331.00000      1.00000      4.00000      0.00000
# min:   5.5000    166.37500      0.01000      0.01000
# max:  16.5000   4492.12500     10.00000     10.00000
#----------
# fit by murnaghan equation equation of state
#----------
#iter, alatt0, vol0 (ula**3),  b0 (mbar),  b0prime,  e0,  sq sum, ierr
# 20  11.0403  1345.68317     0.15379     4.35236   -71.9440324     0.00010 0
#----------
# a (a.u.),   E
#----------
  10.00000        -71.903320  0.040713
  10.01919        -71.905118  0.038914
  10.03838        -71.906862  0.037171
  10.05758        -71.908552  0.035481
  10.07677        -71.910189  0.033843
  10.09596        -71.911775  0.032258
    ⋮
    ⋮
```

```
11.84242        -71.930581  0.013451
11.86162        -71.929995  0.014038
11.88081        -71.929398  0.014634
11.90000        -71.928792  0.015241
```

The output file file.out is suitable to be fed directly to the Xmgr graphics tool. The murn program writes also file called murn.gnu which has the same structure as the numeric table in file.out, but contains *nnn* records, only for the input lattice constants.

## 1.3 Command-line interface: murfit

In principle the information in Sec. 1.2 should allow you to create the required input and run the murn program. This requires, anyway, file editing which some of you may not really appreciate ;-). Sometimes you may also want to add comments in the file where you have stored the originally calculated (*alat*,*eninp*) pairs, etc.

For these and other reasons we provide a simple Bourne-shell script murfit which allows you to create consistent input and run the murn program. Instead of editing files you have full control over the input parameters via command-line options. So, for those of you who are already familiar with the command-line oriented UNIX interface murfit will behave like an ordinary UNIX command.

To start with, you can simply type on the command line murfit and you will get short info about the kind of arguments this command would accept:

```
$ murfit
Usage: murfit [-nuV <number>] [-Ght] [-z [-gTv <number>]] file
```

The general synopsis for usage of the murfit tool is:

**murfit** [OPTION] FILE

The murfit tool first reads in the input data from FILE, which as we shall see bellow have very generic and simple format, brings them to the required format according to the provided command-line options (if supplied) and/or default settings and runs the murn program. The output is stored in file called FILE.murn.

The following options can be given to murfit :

| OPTION | DESCRIPTION |
|---|---|
| -G | murfit reads FILE and creates a temporary file (by adding extension .inp) which has the format already described in Subsec. 1.2.1. Upon completion murfit removes FILE.inp. With this option you can suppress the cleaning of the input file FILE.inp, so that you will be able to inspect it. |
| -g ⟨number⟩ | When requesting zero-point corrections (-z option is set) with this option you can specify value for the Grüneisen constant $\gamma$ (input parameter *gamma*). By default murfit uses the value for Al, $gamma = 2.19$. |

-h Displays detailed help and exits.

-n ⟨number⟩ With this option you can provide value for the *nr_alat* input parameter. By default `murfit` uses $nr_alat = 30$.

-T ⟨number⟩ When requesting zero-point corrections (-z option is set) with this option you can specify the value for the *thetad* input parameter (Debye temperature $\Theta_{D,0}$). By default `murfit` uses the value for Al, $thetad = 428$ K.

-t With this option you enforce `murfit` to create the input file FILE.inp and exit without running the `murn` program.

-u ⟨number⟩ With this option you can specify the *iunut* input parameter. By default `murfit` assumes atomic units for energy, $iunit = 3$.

-V ⟨number⟩ With this option you can specify the value for the *convav* input parameter. By default `murfit` uses $convav = 1.00$.

-v ⟨number⟩ When requesting zero-point corrections (-z option is set) with this option you can specify value for the *atvol* input parameter, $\Omega_0$. By default `murfit` uses the value for Al, $atvol = 109.9$.

-z This option forces `murfit` to include zero-point energy corrections, i.e. setting *tzero*=`.true.` (see also -g, -T, and -v options).

**FORMAT OF** FILE

FILE

```
# This is a comment line.
# another comment line...
a_1     E_1   :lattice constant, total energy
a_2     E_2
 ⋮       ⋮
a_n     E_n
```

**EXAMPLES**

*Example 1*: The following command runs `murfit` in "debug" mode for a system with simple cubic structure and the energy in file is given in atomic units (default settings). The file file.inp will not be removed:

```
$ murfit -G file
```

*Example 2*: This example demonstrates how to perform Murnaghan fit for Al including zero-point energy correction (default setting) if the total energy in file is given in atomic units and face-centered cubic unit cell was used in the calculations:

```
$ murfit -V0.25 -z file
```

*Example 3*: The following example demonstrates how to request a finer mesh to be used by the `murn` program to sample the fitting polynomial, assuming defaults for all other parameters. This will produce output numeric table of 100 rows:

```
$ murfit -n 100 file
```

**SHELL VARIABLES**

Because `murfit` is only a command-line interface to the `murn` program it must know where to find the main program. The path to and the name of the executable generated by compiling murn.f can be specified, respectively, by the shell variables `SOURCEDIR` and `PROGRAM` set in the `murfit` script. Edit the defaults for these variables in order to customize the script for your particular needs. For the example in Sec. 1.2 these would look like: `PROGRAM="murn.x"`, and, assuming that `murn.x` is located in the `/src/murn` subdirectory of your home directory,
`SOURCEDIR="${HOME}/src/murn"`.

**NOTES**

If the `murn` program fails to carry out the least-squares fit to the input data the numeric table in FILE.murn will contain compiler-dependent string identifier (like `NaNQ`) in the energy column. This might indicate "bad" quality of the input data—large scattering in the calculated energies, insufficient number of points, etc.

## 1.4 Example: equilibrium bulk lattice constant of GaAs in LDA

In many cases the first step in given scientific project in computational materials science is to determine the *theoretical* equilibrium lattice constant of the material to be studied. This section demonstrates how this can be accomplished for bulk GaAs using the `fhi98md` program package in conjunction with the `murfit` tool described in the previous section.

In order to prepare input for the `murn`program we need to calculate the total energy of bulk GaAs for certain number of lattice constants. The experimental lattice constant of this compound is $a = 5.653 \ = 10.683$ a.u., and hence we choose 12 values in the range $a \in [9.7, 11.1]$ a.u. to vary the lattice constant in the calculation. It is also advisable to ensure finer sampling of the interval where the minimum is expected.

Below we propose a sample Bourne-shell script that will run the `fhi98md` program for the chosen lattice constants, and prepare input file suitable for the `murfit` tool. The proviso should be made that this solution is not unique in itself and assumes that you work on a UNIX platform.

run.GaAs.sh

```
#!/bin/sh -f

#-----------------------------------------------------
# Setting shell variables
#-----------------------------------------------------
FHI98MD=$HOME/fhi98md
PSEUDO="$HOME/pseudo"
WORK="$HOME/scratch"
RESULT=./

if [ ! -d $WORK ]; then
  mkdir -p $WORK
fi

#-----------------------------------------------------
# Change to working space
#-----------------------------------------------------
cd $WORK

#-----------------------------------------------------
# Copying pseudopotentials to the working space
#-----------------------------------------------------
INX=0
for PSP in ga:lda:ham as:lda:ham
do
  INX=‘expr $INX + 1‘
  cp $PSEUDO/$PSP.cpi fort.1$INX
done

#-----------------------------------------------------
# Copying binaries to the working space
#-----------------------------------------------------
cp $FHI98MD/start/fhi98start .
cp $FHI98MD/fhimd/fhi98md .

#-----------------------------------------------------
# Create inp.mod and constraints.ini
#-----------------------------------------------------
cat > inp.mod << EOF_1
-1    50   14400              : nbeg   iprint timequeue
500   1                      : nomore nomore_init
40.0  0.3                    : delt   gamma
 6.0  0.2  0.0001            : delt2  gamma2 eps_chg_dlt
400   2                      : delt_ion nOrder
0.0   1.0                    : pfft_store mesh_accuracy
2     2                      : idyn i_edyn
0 .false.                    : i_xc   t_postc
.F.   0.001   .F.   0.002    : trane  ampre  tranp amprp
.false. .false. .false. 137  : tfor   tdyn   tsdp nstepe
.false.                      : tdipol
0.00001 0.0005 0.1           : epsel  epsfor epsekinc
0.001   0.001  3             : force_eps max_no_force
3                            : init_basis
EOF_1
#
cat > constraints.ini << EOF_2
0
0
EOF_2

#-----------------------------------------------------
# Set output file identifier
#-----------------------------------------------------
OUTFILE="GaAs_LDA_10Ry_sc"
touch ${RESULT}/${OUTFILE}.dat

#-----------------------------------------------------
# Loop over lattice parameter
#-----------------------------------------------------
for ALAT in 11.1 10.9 10.8 10.7 10.6 10.5 10.4 10.3 10.2 10.1 10.0 9.7
do
```

```
#-----------------------------------------------------
# Create start.inp
#-----------------------------------------------------
cat > start.inp << EOF_3
 1                    : number of processors
 1                    : number of minimal processors
 1                    : number of processors per group
 2                    : number of species
 0                    : excess electrons
 5                    : number of empty states
1  0                  : ibrav pgind
${ALAT} 0.0 0.0 0 0 0 : celldm
 1                    : number of k-points
 0.5  0.5  0.5  1.0   : k-point coordintes, weight
 4 4 4                : fold parameter
 .true.               : k-point coordinates relative or absolut?
10.0  2.0             : Ecut [Ry], Ecuti [Ry]
 0.01  .true. .f.     : ekt tmetal tdegen
 .true. .false.  1    : tmold tband nrho
 5  2  1234           : npos  nthm  nseed
873.0 1400.0 1e8 1    : T_ion T_init Q nfi_rescale
 .true. .true.        : tpsmesh coordwave
4  3 'Gallium'        : number of atoms, zv, name
1.0  69.72  0.7 3 3   : gauss radius, mass, damping,l_max,l_loc
.t. .t. .f.           : t_init_basis
0.0 0.0 0.0     .t.   : tau0 tford
0.5 0.5 0.0     .t.   : tau0 tford
0.5 0.0 0.5     .t.   : tau0 tford
0.0 0.5 0.5     .t.   : tau0 tford
4 5 'Arsenic'         : number of atoms, zv, name
1.0 74.92  0.7 3 3    : gauss radius, mass, damping, l_max,l_loc
.t. .t. .f.           : t_init_basis
0.25 0.25 0.25  .t.   : tau0 tford
0.75 0.25 0.75  .t.   : tau0 tford
0.75 0.75 0.25  .t.   : tau0 tford
0.25 0.75 0.75  .t.   : tau0 tford
EOF_3

#-----------------------------------------------------
# Run fhi98start utility to prepare input to fhi98md
#-----------------------------------------------------
  ./fhi98start | tee start.out

#-----------------------------------------------------
# Run fhi98md program
#-----------------------------------------------------
  ./fhi98md

#-----------------------------------------------------
# Extract total energy from fort.6
#-----------------------------------------------------
  ETOT=‘
    grep ’^>>>[ ,0-9]’ fort.6 |
    tail -1l                  |
    sed -e ’1,$s/>>>//’       |
    awk ’{print $6}’‘

#-----------------------------------------------------
# Write lattice constant and corresponding total energy
# in format suitable for murfit
#-----------------------------------------------------
  echo "${ALAT}  ${ETOT}" >> ${RESULT}/${OUTFILE}.dat

#-----------------------------------------------------
# Store output file fort.6
#-----------------------------------------------------
  cp fort.6 ${RESULT}/fort.6-${OUTFILE}.a=${ALAT}
done #--- end of loop over lattice parameter
```

If you make the run.GaAs.sh script executable then you can simply run it interactively:

```
$ ./run.GaAs.sh &
```

After finishing you will have a file GaAs_LDA_10Ry_sc.dat that contains the input need to perform the Murnaghan fit with `murfit`. We have added also some comments to remind us about the most important settings used to obtain the data.

GaAs_LDA_10Ry_sc.dat

```
#--- GaAs (sc) ---
# LDA, Ecut=10 Ry
# k-mesh: 4 x 4 x 4
# plain pseudopotentials
#--- 08.06.2001 ---
11.1  -34.49958
10.9  -34.51772
10.8  -34.52467
10.7  -34.53067
10.6  -34.53577
10.5  -34.53779
10.4  -34.53622
10.3  -34.53481
10.2  -34.52664
10.1  -34.51926
10.0  -34.50892
9.7  -34.44377
```

With this you have almost completed your job. Now you supply GaAs_LDA_10Ry_sc.dat to the `murfit` tool as already explained in Sec. 1.3. As we have used simple-cubic unit cell for GaAs and the total energy is given in a.u. we can stay with the default settings employed by `murfit`; only a finer mesh is requested for the fitting polynomial in order to get very smooth curve for plotting:

```
$ murfit -n 50 GaAs_LDA_10Ry_sc.dat
```

The result from the fitting is stored in the GaAs_LDA_10Ry_sc.dat.murn file in the current working directory. The equilibrium lattice constant $a_0$, bulk modulus $B_0$, and its pressure derivative $B_0'$ for GaAs within the local density approximation and 10 Ry cutoff energy read, respectively,

$$a_0 = 10.4590 \text{ a.u.}, \qquad B_0 = 0.71921 \text{ mbar}, \qquad B_0' = 5.09958.$$

GaAs_LDA_10Ry_sc.dat.murn

```
# unit of length =       0.529177  angstroem
# energies input in HARTREES
# unit cell volume = a**3 *       1.000000
# a(min)=           9.7000000
# a(max)=          11.1000000
# points:  50
# number of data points: n =   12
#----------
#    i     alatt         volume              E
#     1       9.700000     912.673000     -34.4437700000
#     2      10.000000    1000.000000     -34.5089200000
#     3      10.100000    1030.301000     -34.5192600000
#     4      10.200000    1061.208000     -34.5266400000
```

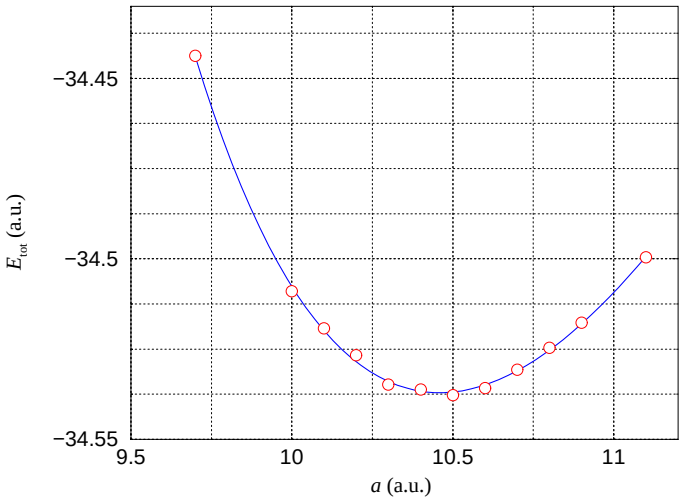



Figure 1.1: Murnaghan fit (full curve) to the calculated total energy (circles) of bulk GaAs as a function of the lattice constant in local-density approximation, and 10 Ry cutoff energy for the plane-waves basis set. A simple cubic unit cell ($ibrav = 1$) comprising 8 atoms was used to represent the bulk material (cf. run.GaAs.sh). Brillouin zone sampling employs the Monckhorst-Pack set with single **k**-point ($nkpt =$ `1`, $xk =$ (`0.5 0.5 0.5`), $wkpt =$ `1.0`) and $i_facs = 4 \times 4 \times 4$ folding factor.

```
#      5      10.300000    1092.727000     -34.5348100000
#      6      10.400000    1124.864000     -34.5362200000
#      7      10.500000    1157.625000     -34.5377900000
#      8      10.600000    1191.016000     -34.5357700000
#      9      10.700000    1225.043000     -34.5306700000
#     10      10.800000    1259.712000     -34.5246700000
#     11      10.900000    1295.029000     -34.5177200000
#     12      11.100000    1367.631000     -34.4995800000
#----------
#      alatt0    vol0 (ula**3)  b0 (mbar)     b0prime        e0
#    10.5000    1157.62500      1.00000      4.00000      0.00000
# min:    5.2500     144.70312      0.01000      0.01000
# max:   15.7500    3906.98437     10.00000     10.00000
#----------
# fit by murnaghan equation equation of state
#----------
#iter, alatt0, vol0 (ula**3),  b0 (mbar),  b0prime,  e0,  sq sum, ierr
#  20  10.4590  1144.10406     0.71921     5.09958   -34.5370685     0.00053 0
#----------
# a (a.u.),   E
#----------
   9.70000        -34.443901  0.093167
   9.72857        -34.452028  0.085040
     :
     :
  11.07143        -34.502306  0.034762
  11.10000        -34.499330  0.037739
```

Sometimes you may want to have the bulk modulus $B_0$ measured in GPa, then you have to simply multiply the corresponding field from the output file ....murn by 100; in this particular case $B_0 = 71.921$ GPa.

Visualization of the Murnaghan fit and the calculated points is especially

simple with the Xmgr graphics tool as this software will skip the #-commented lines. You can first read in the Murnaghan fit GaAs_LDA_10Ry_sc.dat.murn, and then superimpose the calculated points from GaAs_LDA_10Ry_sc.dat. Sample plot is given in Fig. 1.1, where we have played somewhat with the Xmgr plotting options.

It is important to notice that there is some (although very small) scattering in the calculated points. This feature is related to the fact that we used the same cutoff of 10 Ry to calculate total energies for varying supercell volume. We have therefore used plane-wave basis sets with different quality in each of the 12 runs, because of the volume dependence of the mean number of plane waves $\bar{N}_{\mathrm{PW}}$ used to expand the Kohn-Sham orbitals. While for high energy cutoffs this effect would not affect the Murnaghan fit, for low cutoffs large scattering in the calculated points could significantly bias the fitting procedure or eventually make it impossible.

In the latter cases it is therefore imperative to first correct the calculated total energies and only then perform the Murnaghan fitting. Commonly used schemes for introducing corrections to total energy and pressure are, for example, those by Francis and Payne [6], and the scaling hypothesis due to Rignanese and co-workers [7].

# Chapter 2

# `Contour Plot` Package

## 2.1 Contour Plots

Often one needs information about the character of the chemical bonds between atoms in a solid, molecule or surface. This can be obtained by analyzing the charge density of the system (or charge density differences between different systems). In this section we will show how to create contour plots of the total charge density, charge density differences and charge density for a single state and how to simulate scanning tunneling microscopy (STM) images using the `Contour Plot` package. The `Contour Plot` package is a separate program that reads the output files of the `fhi98md` and projects the charge density on a given plane.

### 2.1.1 STM simulation

Tersoff and Hamann [8, 9] showed that the tunneling current in STM can be approximated by the expression

$$I = \frac{2\pi e}{\hbar} \sum_{i,k} f(E_i)(1 - f(E_k + eV))|M_{i,k}|^2 \delta(E_k - E_i). \tag{2.1}$$

Here, $M_{i,k}$ is the tunneling matrix element between states $\Psi_i$ of the probe and $\Psi_k$ of the surface, $V$ is the voltage applied between tip and surface, and $f$ is the Fermi function. For low enough temperatures, the term $f(E_i)(1 - f(E_k + eV))$ is non-zero only if $E_k$ lies in a narrow interval of energies. If we further assume that $|M_{i,k}|^2 \propto |\Psi_k|^2$ in this energy range, the tunneling current is proportional to the density of states of the surface, summed up over the energy interval. The `Contour Plot` package takes the states that contribute to the STM image, within this energy interval, and calculates the density due to those states. For metals, typically very small voltages are applied, and one can take the tunneling current to be proportional to the density of states of the surface at the Fermi level.

A detailed description of the functionalities implemented in the package is given in the next section.

### 2.1.2 Total charge density and charge density difference

To calculate the total charge density, a sum over all occupied states is performed,

$$\rho(\mathbf{r}) = \sum_{i,k}^{\text{occ}} |\Psi_{i,k}(\mathbf{r})|^2 \tag{2.2}$$

This quantity is stored in the fort.72 file in binary format. The file is one of the output files of the `fhi98md`.

Suppose that you want to calculate the charge density difference between different systems. The charge density difference can be calculated as:

$$\Delta\rho(\mathbf{r}) = \rho(\mathbf{r})^{\text{I}} - \rho(\mathbf{r})^{\text{II}} - \rho(\mathbf{r})^{\text{III}} \tag{2.3}$$

where the systems I, II and III can be, for example, the charge density of the adsorbate+substrate, the substrate alone, and the isolated adsorbate atom, respectively.

### 2.1.3 Single state

A single state $\Psi_{i,k}(\mathbf{r})$ is selected be specifying its **k**-point and band indices, $k$ and $i$, respectively. To calculate the corresponding density,

$$\rho(\mathbf{r})^{\text{single}} = |\Psi_{i,k}(\mathbf{r})|^2, \tag{2.4}$$

the fort.71 binary file (generated by the `fhi98md` program) is used.

## 2.2 Input file

### 2.2.1 Format

The input file needed for the execution of the `Contour Plot` program is called contour.inp and its format is shown below.

contour.inp

```
&planes                              : <--- assignments for planes ---
plane%x_axis = 0,0,-0.386            : 1st vector in a.u.
plane%y_axis = 2,-2,0                : 2nd vector in a.u.
plane%origin =0,-0,-0.182            : origin of the vectors defined above
plane%mesh =  200,200                : mesh of points
plane%name = 'lattice'               : unit of the coordinates above
/                                    : <---
'options'                            : stm, single or total
l1,l2,l3                             : rho_mesh (nnr1+1, nnr2, nnr3)
omega                                : volume of the unit cell in a.u.
i_select                             : selected state
ik_select                            : selected k-point
u_min                                : minimum energy for STM simulation in eV
u_max                                : maximum energy for STM simulation in eV
  2.806507   -4.861012    0.000000  : a1
  2.806507    4.861012    0.000000  : a2
  0.000000    0.000000   50.413071  : a3
  1.414214   -0.816497    0.000000  : b1
  1.414214    0.816497   -0.000000  : b2
 -0.000000    0.000000    0.157459  : b3
6                                    : nrot
1-----------------------             : matrix rotation
   1        0        0
```

```
   0        1        0
   0        0        1
.
.
.

6---------------------
  -1        1        0
  -1        0        0
   0        0        1
                              : blank line
 3            8               : number of k-points, number of states
 1            1            1  : 1st column: enumerate the k-point
 1            2            1    2nd column: enumerate the state of the n-kpoint
 1            3            1    3rd column: if 1, the state contributes to
 1            4            1                      the summation in Eq. (2.1)
 1            5            1                if 0, the state does not contribute to the
 1            6            0                      summation in Eq. (2.1)
 1            7            0
 1            8            0
 2            1            1
 2            2            1
 2            3            1
 2            4            1
 2            5            0
 2            6            0
 2            7            0
 2            8            0
 3            1            1
 3            2            1
 3            3            1
 3            4            1
 3            5            0
 3            6            0
 3            7            0
 3            8            0
```

Note that the relative coordinates of the vectors and the origin of the vectors as well should be negative, to be consistent with the implementation of the `fhi98md` code.

### 2.2.2 Defining the plane and the range of the contour plot

In the first six lines of the input file contour.inp you have to provide the variables that define the plane where your charge density is projected onto. Two vectors have to be provided to define this plane (lines 2 and 3) and the origin of those vectors (line 4). Line 5 defines the 2D-mesh of the plot. Note that the larger the number of points, the better is the resolution of your plot.

| Record | Parameter | Kind | description |
|---|---|---|---|
| 1 | *planes* | | dummy argument |
| 2 | *plane%x_axis* | 3× real | 1st vector in units of $(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$ |
| 3 | *plane%y_axis* | 3× real | 2nd vector in units of $(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$ |
| 4 | *plane%origin* | 3× real | Origin of the vectors defined above |

| Record | Parameter | Kind | description |
|---|---|---|---|
| 5 | *plane%mesh* | 2× integer | Number of mesh points. The number of points should have the correct aspect ratio. For example, if you define $\|plane\%x_axis\| = 1$ and $\|plane\%y_axis\| = \sqrt{2}$, then you have to set for example *plane%mesh*=100, 141. |
| 6 | *plane%name* | character | Units of the variables above described (only tested for option *lattice*). |

### 2.2.3 Setting the plotting options

Here we explain how to set up the proper variables in the contour.inp file in order to simulate STM images, total charge density, difference of charge density or charge density for a single state. Depending on the chosen *option* you have to provide either the binary file fort.72, which contains the charge density, or fort.71, which contains the wave function. Additionally you have to provide the contour.inp file. A short description of the options is given below.

| Record | Parameter | Kind | description |
|---|---|---|---|
| 7 | '*total*' | character | Calculates the total charge density reading the binary file fort.72. The output file is contour.dat. |
| | '*single*' | | Calculates the charge density for a specific k-point and state. The wave function (file fort.71) is read in and the output file is contour.dat. |
| | '*STM*' | | Simulates STM images. The input file fort.71 is read in and output file is contour.dat. |
| | '*diff*' | | Calculates the contour plot of charge density and charge density difference.<br>In this case, three input files should be read in. For example:<br>fort.72: charge density of the complete system I<br>fort.73: charge density of subsystem II<br>fort.74: charge density of the subsystem III<br>If you use this option, rename the fort.72 files generated by the `fhi98md` code according to the names given above.<br>The output files containing the contour plot lines are:<br>contour1.dat: 2D charge density of the complete system<br>contour2.dat: 2D charge density of the system II<br>contour3.dat: 2D charge density of the system III<br>charge_diff.out: 2D charge density difference |

### STM simulation

You can simulate STM images (empty and filled states). As already mentioned, the wave function generated by the `fhi98md` code is stored in the binary file fort.71. If you want to simulate STM images (filled or empty states), you should provide the file `fort.71` together with the file contour.inp. The variable *options* in the contour.inp file should be set equal to `STM`. Additionally you have to provide the states which contribute for the STM simulation.

### Total Charge Density

To obtain the contour lines, you have to edit the input file `contour.inp` to set up the variable *options* = `total`. The charge density generated by the `fhi98md` package is stored in binary file fort.72. The contour program reads the `fort.72` file, produces the charge in real space and projects the charge density onto the plane you defined in contour.inp.

### Charge Density Difference

In this case you have to edit the input file contour.inp to set up the variable *options* = `diff`. As input files you have to provide the charge densities (binary files fort.72) of the three systems ($\rho^{\mathrm{I}}$, $\rho^{\mathrm{II}}$, $\rho^{\mathrm{III}}$) from which you want the charge density difference.

### Single state

To use this option edit the contour.inp file and provide the k-point (*ik_select)* and band index (*i_select*) of the state you are interested in, like defect or surface states for example. Further set *options* to `single`. To choose the **k**-point and the state you are interested in, take a look at the output file fort.6. For example, if you want the first **k**-point and the tenth state, you should set *ik_select* = 1 and *i_select* = 10.

| Record | Parameter | Kind | description |
|---|---|---|---|
| 8 | *l1+1,l2,l3* | integer | Number of mesh points in the `fort.72` file. Note that the first component of the mesh should be increased by 1 unit. |
| 9 | *omega* | real | Volume of the unit cell of the system. |
| 10 | *i_select* | integer | Selected state in case of option *single*. For the other options, it is not relevant. |
| 11 | *ik_select* | integer | Select k-point in case of option *single*. For the other options, it is not relevant. |
| 12 | *u_min* | real | Minimum energy in case of option *stm*. This defines the lowest state that should be included in the sum (see Eq. (2.1)). For the other options, it is not relevant. Only needed if you use the shell script described in the next section. |

| Record | Parameter | Kind | description |
|---|---|---|---|
| 13 | *u_max* | real | Maximum energy in case of option *stm*. This defines the highest state that should be included in the sum (see Eq. (2.1)). For the other options, it is not relevant. u_min and u_max define the number of states that should be included in the STM simulation. Only needed if you use the shell script described in the next section. |
| 14-16 | $\mathbf{a}_1$,$\mathbf{a}_2$,$\mathbf{a}_3$ | real | Primitive lattice vectors in a.u. |
| 17-19 | $\mathbf{b}_1$,$\mathbf{b}_2$,$\mathbf{b}_3$ | real | Reciprocal primitive vectors in units of $2\pi/a$. |
| 20 | *nrot* | integer | Number of symmetry operations, needed in case of option *stm*. |
| | | integer | Matrix with the symmetry operations in the format shown above. |
| three-column list | | | |
| | *i_state* | integer | band index state in case of option *stm*. For the other options, it is not relevant. |
| | *i_k* | integer | k-point in case of option *stm*. For the other options, it is not relevant. |
| | *focc* | real | If `0`, set the occupation number of a specific state equal to `0` and if `1` set the occupation number equal to `2` in case of *option* = *stm*. For the other options, it is not relevant. |

## 2.3 Output

### 2.3.1 Format

Depending on the *option* you have chosen in the input file `contour.inp` the `Contour Plot` program generates different output files. We give a brief description of them below. The general format is the following:

contour.dat

```
165  50
    0.0006892362725
    0.0005351907358
    0.0004128452768
    0.0003171773621
    0.0002425657761
    0.0001841769710
    0.0001388096542
     :
     :
```

| Field | Parameter | Kind | description |
|---|---|---|---|
| 1 | *plane%mesh* | 2×integer | 2D mesh size defined in the input file |
| 2 | $\rho$ | real | charge density $\rho$ along the plane defined in contour.inp file. The unit of charge density is $Bohr^{-3}$ |

### 2.3.2 STM simulation, charge density, and single state

The output file generated in this case is contour.dat.

### 2.3.3 Charge density difference

In this case the files contour1.dat, contour2.dat and contour3.dat are generated, the charge density of the corresponding systems from which you calculate the difference. The charge density difference, Eq. (2.3), is given in the file charge_diff.out.

## 2.4 Shell script

In order to make life easier we propose the shell script `run.sh` described below. This script compiles the package, generates the file contour.inp, runs the program, generates the output files, and draws the contour plot. To do this, just type:

```
$ run.sh
```

Please make sure that the file is executable. The basic idea of the shell script is that you have only to provide those input data that are not provided by the output files of the `fhi98md` code. Those variables are extracted from the output files fort.6, report.txt and input file inp.ini. In addition, the shell script defines the paths of the required input files, the working directory and the source program. You should adapt the variables according to your directory structure. For example, if the `contour.exe` is located in the subdirectory `Dcontour` of your home directory, then set `CONTOUR = "$HOME/Dcontour"`.

Below we give an example of how to calculate the total charge density of GaAs bulk , assuming you have already created all required files and run the `fhi98md` program. In this example, the lattice parameter used to get a converged charge density and wave function was $a_{lat} = 10.683$ a.u.. The Brillouin zone was sampled using a Monkhorst-Pack k-point mesh 4×4×4 and the energy cutoff was 15 Ry.

run.GaAs.total.sh

```
#!/bin/csh -f

#-----------------------------------------------------------------
# Setting directories
#-----------------------------------------------------------------

set RHO = "$HOME/Drho"
```

```
set PSI = "$HOME/Dpsi"
set CONTOUR = "$HOME/Dcontour"
set WORK = "$HOME/Dwork"

#-----------------------------------------------------------------
# Setting variables
#-----------------------------------------------------------------

set INPUT = contour.inp

#-----------------------------------------------------------------
#Compiling the Contour Plot package
#-----------------------------------------------------------------

cd $CONTOUR

#./run.compilation

#-----------------------------------------------------------------
#Copying the executable to the working directory
#-----------------------------------------------------------------

cp $CONTOUR/contour.exe  $WORK

#-----------------------------------------------------------------
#Creating input file contour.inp
#-----------------------------------------------------------------

cd $WORK

cat > contour.inp << END_1
&planes
plane%x_axis = 0,0,-0.386
plane%y_axis = 2,-2,0
plane%origin =0,-0,-0.182
plane%mesh =  200,200
plane%name = 'lattice'
/
'total' : options
END_1
#-------------------------------------------------------------
# setting variables not extracted from the output files
# you have to edit here according to your wishes
#-------------------------------------------------------------

@ U_MIN = -10
@ U_MAX = 4
@ I_SELECT = 1
@ IK_SELECT = 1

#-------------------------------------------------------------
# setting variables extracted from the fort.6, inp.ini and
# report.txt files
#-------------------------------------------------------------

set MESH = `grep 'mesh= ' $WORK/fort.6 | awk '{print $12, $13, $14}'`
echo "$MESH : mesh" >> $WORK/$INPUT
set OMEGA = `grep 'omega=' $WORK/fort.6 | awk '{print $10}'`
echo "$OMEGA : omega" >> $WORK/$INPUT

#-------------------------------------------------------------

echo "$I_SELECT : i_select" >> $WORK/$INPUT
echo "$IK_SELECT : ik_select" >> $WORK/$INPUT
echo "$U_MIN : u_min"  >> $WORK/$INPUT
echo "$U_MAX : u_max" >> $WORK/$INPUT

#-------------------------------------------------------------
# setting the primitive lattice vectors
#-------------------------------------------------------------

set A1 = `grep 'a1'  $WORK/fort.6 | awk '{print $2,  $3, $4}'`
echo "$A1 : a1"  >> $WORK/$INPUT
set A2 = `grep 'a2'  $WORK/fort.6 | awk '{print $2,  $3, $4}'`
```

```
echo "$A2 : a2" >> $WORK/$INPUT
set A3 = `grep 'a3'  $WORK/fort.6 | awk '{print $2,  $3, $4}'`
echo "$A3 : a3" >> $WORK/$INPUT

#-------------------------------------------------------------
# setting the reciprocal lattice vectors
#-------------------------------------------------------------

set B1 = `grep 'b1'  $WORK/fort.6 | awk '{print $2,  $3, $4}'`
echo "$B1 : b1" >> $WORK/$INPUT
set B2 = `grep 'b2'  $WORK/fort.6 | awk '{print $2,  $3, $4}'`
echo "$B2 : b2" >> $WORK/$INPUT
set B3 = `grep 'b3'  $WORK/fort.6 | awk '{print $2,  $3, $4}'`
echo "$B3 : b3" >> $WORK/$INPUT
set NROT = `grep 'nrot' $WORK/inp.ini | awk '{print $1}'`

#-------------------------------------------------------------
# setting the operation symmetries
#-------------------------------------------------------------

echo "$NROT : nrot" >> $WORK/$INPUT
awk -f sym.awk $WORK/inp.ini >> $WORK/$INPUT

# add a dummy line
echo "" >> $WORK/$INPUT

#-------------------------------------------------------------------
# building up the contour.inp according to OPTIONS:
# STM, total, diff or single
#-------------------------------------------------------------------

set OPTIONS = `grep 'options' $WORK/contour.inp |awk '{print $1}'`

if ($OPTIONS == "'STM'") then
  set NKPT = `grep ': nkpt' $WORK/inp.ini | awk '{print $1}'`
  set NSTATES = `grep 'nx,' $WORK/inp.ini | awk '{print $3}'`
  set U_MIN = `grep 'u_min' $WORK/$INPUT | awk '{print $1}'`
  set U_MAX = `grep 'u_max' $WORK/$INPUT | awk '{print $1}'`
  echo "$NKPT $NSTATES : nkpt, nstates"  >> $WORK/$INPUT
#-------------------------------------------------------------------
  set FILE = report.temp

  awk -f eig.awk $WORK/report.txt > ${FILE}

  set INKPT=1
  set START=1
  set END=${NSTATES}

  while ( $INKPT <= ${NKPT} )
    sed "${START},${END}\!d" ${FILE} |\
    awk -v e_min=${U_MIN} -v e_max=${U_MAX} -v kpt=${INKPT} '{\
      if ( e_min <= $1 && $1 <= e_max )\
        focc = 1.0\
      else\
        focc = 0.0\
      printf("%3d  %3d   %5.3lf %s\n", kpt, NR, focc, ": kpt n occ");\
    }' >> $WORK/contour.inp
    @ INKPT++
    @ START = $END + 1
    @ END = $INKPT * ${NSTATES}
  end
else
  echo "Nothing to be done..."
endif

#-----------------------------------------------------------------
# Setting the path to the files fort.71 or fort.72
#-----------------------------------------------------------------

  if ("$OPTIONS" == "'single'" || "$OPTIONS" == "'STM'" ) then
         ln -s $PSI/fort.71 $WORK/fort.70
  elseif ("$OPTIONS" == "'total'" || "$OPTIONS" == "'diff'") then
         ln -s $RHO/fort.72 $WORK/fort.72
  endif
```

```
#-------------------------------------------------------------------
# Running the contour.exe
#-------------------------------------------------------------------

${CONTOUR}/contour.exe

#-------------------------------------------------------------------
#Ploting the contour lines using gnuplot

run.plot
```

## 2.5 Examples

### 2.5.1 Example 1: total charge density for GaAs bulk

As an example we consider a total energy calculation for GaAs bulk. The Monkhorst-Pack k-point mesh is $4\times4\times4$ with the initial k-point $(0.25, 0.25, 0.25)$ and an energy cutoff of 15 Ry was used. Here a $350 \times 494$ mesh of points is used. The result is shown in Fig. 2.1.

### 2.5.2 Example 2: charge density for a specific state of GaAs bulk

You can be also interested in calculating the total charge density and wave function for a specific k-point and state in real space. In this example we show a plot of charge density for GaAs bulk. We take the highest occupied state at the $\Gamma$ point. The major contribution comes from the As atom, which we can easily see comparing the Fig. 2.1 and Fig. 2.2. The result is shown in Fig. 2.2. Here a $200 \times 282$ mesh of points is used.

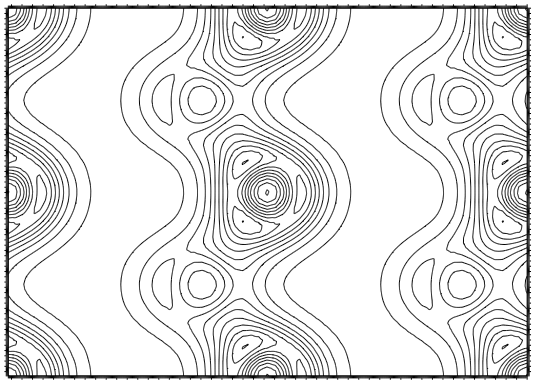


Figure 2.1: Valence charge density contour plots of GaAs bulk in the zinc-blende structure (2 atoms per unit cell) within LDA.

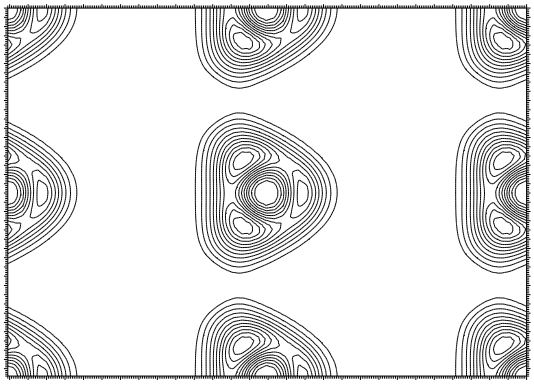


Figure 2.2: Valence charge density contour plots of GaAs bulk within LDA. The plot show the contour lines at the Γ point for the highest occupied state.

### 2.5.3 Example 3: STM simulation for GaAs (110)

The Monkhorst-Pack k-points mesh was taken as $4 \times 4 \times 1$ with the initial k-point $(0.25, 0.25, 0)$ and an energy cutoff of 15 Ry was used. Here a $(300 \times 423)$ mesh of points is used. The result is shown in Fig. 2.3.

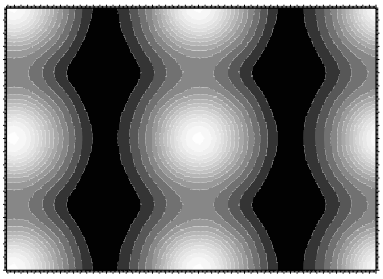
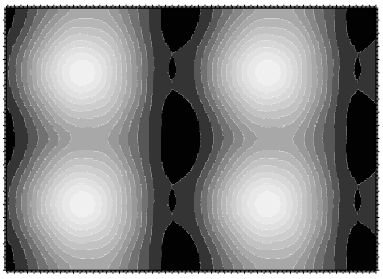
(a)  (b) 

Figure 2.3: STM simulation for GaAs (110) surface within LDA. The upper plots show (a) filled states (As dangling bond) and (b) empty states (Ga dangling bond). The surface tip separation was taken as 3 Å.

# Chapter 3

# Projected Density of States

## 3.1 Some definitions

The projected density of states (PDOS) is calculated by projecting the localized atomic orbitals on the Kohn-Sham states $\Psi_{ik}(\mathbf{r})$:

$$g_\mu(\epsilon) = \sum_k w_k \sum_i \mathcal{F}(\epsilon - \epsilon_{i,k}) \left| \left\langle \phi_{k,\mu}^{w_\mu} \left| \frac{1}{e^{(r-d_\mu)/w_\mu}+1} \right| \Psi_{i,k}(\mathbf{r}) \right\rangle \right|^2 \tag{3.1}$$

where $\mathcal{F}(\epsilon)$ is a broadening function necessary due to the discreteness of the eigenvalues. We use a Gaussian function

$$\mathcal{F}(\epsilon) = \frac{1}{\sqrt{\pi}\omega} \exp(-\epsilon/\omega)^2, \tag{3.2}$$

or a Lorentzian function

$$\mathcal{F}(\epsilon) = \frac{\omega}{\pi} \frac{1}{\epsilon^2+\omega^2} \tag{3.3}$$

to interpolate the energies.

In Eq. (3.1) $w_k$ is the weight associated with the specific $\mathbf{k}$-point, and $\phi_{k,\mu}^{w_\mu}$ is the atomic pseudo wave function. The radial weight factor

$$\frac{1}{e^{(r-d_\mu)/w_\mu}+1}$$

ensures the vanishong of radial function at a certain distance, so as to avoid overlap between the neighbouring atoms.

## 3.2 Shell script

Below we provide a sample shell script to calculate the PDOS. In this particular case, the PDOS is calculated for GaAs bulk. The shell script runs the `fhi98start` utility, the `fhi98md` program and compiles the utility `dos.f` needed to calculate the projected and total density of states. To visualize the DOS and PDOS, one can use for example the shareware plotting tools `Xmgr` or `Grace`.

run.GaAs.pdos.sh

```
#-------------------------------------------------------------
# setting directories
#-------------------------------------------------------------

set SOURCE="$HOME/Dsource"
set PSEUDO="$HOME/Dpseudo"
set RESULT="$HOME/Dresult"
set WORK="$HOME/Dwork"

#-------------------------------------------------------------
# setting variables
#-------------------------------------------------------------

set POTENTIALS="ga.lda as.lda"

if [ ! -d ${RESULT} ]; then
  mkdir -p ${RESULT}
fi
if [ ! -d ${WORK} ]; then
  mkdir -p ${WORK}
fi

cd ${WORK}

#-------------------------------------------------------------
# creating inp.mod
#-------------------------------------------------------------

cat > inp.mod << EOF_INPMOD
-1 100 100000               : nbeg   iprint timequeue
100 1                       : nomore nomore_init
40.0  0.2                   : delt  gamma
 8.0  0.2  0.0001           : delt2 gamma2 eps_chg_dlt
400 2                       : delt_ion nOrder
0.0 1.0                     : pfft_store mesh_accuracy
2 2                         : idyn i_edyn
0  .false.                  : i_xc t_postc
.false.  0.001  .false. 0.002  : trane ampre tranp amprp
.false. .false. .false. 1800   : tfor tdyn tsdp nstepe
.false.                     : tdipol
0.00001 0.0005 0.1          : epsel epsfor epsekinc
0.001 0.001 3               : force_eps max_no_force
1                           : init_basis
EOF_INPMOD

#-------------------------------------------------------------
# creating start.inp
#-------------------------------------------------------------

cat > start.inp << EOF_STARTINP
 1                  : number of PE's
 1                  : number of minimal PE's
 1                  : number of PE's per Group
 2                  : number of species
 0                  : excess electrons
 4                  : number of empty states
 2  0               : ibrav pgind
 10.47 0.0 0.0 0 0 0 : celldm
 1                  : number of k-points
 0.5  0.5  0.5  1.0  : k-point coordinates, weight
 4 4 4              : fold parameter
 .true.             : k-point coordinates relative or absolut?
 10.0  4.0          : Ecut [Ry], Ecuti [Ry]
 0.004 .true. .false. : ekt tmetal tdegen
 .true. .false. 1    : tmold tband nrho
 5  2  1234         : npos nthm nseed (if tdyn = .true.)
 873.0 1400.0 1e8 1  : T_ion T_init Q nfi_rescale (if tdyn = .true.)
 .true. .TRUE.      : tpsmesh coordwave
 1 3 'Gallium'      : number of atoms, zv, name
 1.0 69.72 0.7 3 3    : gauss radius, mass, damping,l_max,l_loc
 .t. .t. .f.        : t_init_basis
```

```
 0.0 0.0 0.0 f
 1 5 'Arsenic'      : number of atoms, zv, name
 1.0 74.92 0.7 3 3     : gauss radius, mass, damping, l_max,l_loc
 .t. .t. .f.         : t_init_basis
   0.25 0.25 0.25 F
EOF_STARTINP

#-----------------------------------------------------------------------
# creating constraints.ini
#-----------------------------------------------------------------------

cat > constraints.ini << EOF_CONSTR
0
0
EOF_CONSTR

#-----------------------------------------------------------------------
# creating pdos.inp necessary for PDOS
#-----------------------------------------------------------------------

cat > pdos.inp << END_PDOS
.true.
3 3              ! highest pseudopotential component to be included in the PDOS
.true.           ! if .true. k-point inversion symmetry is included
2.2665   0.1     ! cutoff radius  (if  < 0: no cutoff for atomic wavefunctions)
2.2665   0.1     ! cutoff radius  (if  < 0: no cutoff for atomic wavefunctions)
END_PDOS

#-----------------------------------------------------------------------
# moving pseudopotentials and executables
# to the working space
#-----------------------------------------------------------------------

INX=0
for SP in ${POTENTIALS}
do
  (( INX = INX + 1 ))
  cp ${PSEUDO}/${SP}.cpi  fort.1${INX}
done

#-----------------------------------------------------------------------
# running the fhi98start utility and the fhi98md program
#-----------------------------------------------------------------------

cp ${SOURCE}/fhi98start .
cp ${SOURCE}/fhi98md .

fhi98start
fhi98md

#-----------------------------------------------------------------------
# copying the result of the fhi98md run
#-----------------------------------------------------------------------

cp fort.6 coord.out energy fort.71 ${RESULT}

#-----------------------------------------------------------------------
# compiling and running the dos.f program
#-----------------------------------------------------------------------

f77 -o dos.x dos.f
dos.x

#-----------------------------------------------------------------------
# copying the result of the dos.f program
#-----------------------------------------------------------------------

cp pdos-s pdos-p pdos-d pdos-spd tdos ${RESULT}
```

### 3.2.1 Input

During the run of the `fhi98md` program, one has the option of calculating the PDOS or not. In order to calculate the PDOS, it is necessary to provide an additional input file pdos.inp. This file is meant to provide information about the cutoff radii of the wave functions and specify which components of the pseudo wave functions to be used. Here is a short description of the input file format.

pdos.inp

```
.true.          : if .true. calculates PDOS
3 3             : highest pseudopotential component for each specie to be included in the PDOS
.true.          : if .true., includes k-point inversion symmetry
2.2665   0.1    : cutoff radius, width  (if < 0 : no cutoff for pseudo wavefunction)
2.2665   0.1    : cutoff radius, width (if < 0 : no cutoff for pseudo wavefunction)
```

| Rocord | Parameter | Kind | description |
|---|---|---|---|
| 1 | *tf_pdos* | logical | If `.true.`, calculates the PDOS. |
| 2 | *l_read* | integer | The highest pseudopotential angular moment to be included in the PDOS calculation: $s = 1$, $p = 2$, $d = 3$. |
| 3 | *tuseinv* | logical | If `.true.`, inversion symmetry is applied. |
| 4 | *dist, width* | real | Cutoff radius $d_\mu$ and the width $\omega_\mu$ of the pseudo wave function, cf. Eq. (3.1). |

To do the analysis of the PDOS and DOS one needs to run a separate program called dos.f. The files control.inp, report.txt and project.txt have to be provided. The latter two files are generated by the `fhi98md` package and will not be described here (see the `fhi98md` user's guide [2]). The description of the control.inp file is summarized below.

*control.inp*

```
2 1 1 0 0  : total number of species, number of the atom of the (1st,2nd,3rd,4th) species
1 1        : nth atom of the mth species, (e.g., 1, 2 -> 1st atom of the 2nd species)
0.15 1     : broadening factor, 1: Gaussian, 2: Lorentzian
```

| Record | Parameter | Kind | description |
|---|---|---|---|
| 1 | *nsx, nax* | integer | Total number of species, number of atoms of the 1st, 2nd,... *nsx*th species. |
| 2 | *iia, iis* | 2×integer | nth atom of mth species onto which the DOS is projected. For example: `1, 2` means 1st atom of the 2nd species |
| 3 | *width, typ* | 2×integer | Broadening factor. `1`: Gaussian function, `2`: Lorenztian function. |

Note that one atom per run can be specified in control.inp.

### 3.2.2 Output

The program dos.f calculates the total density of states and reads the project.txt file to write the different projections of PDOS. When you run the program dos.f the following output files are produced:

| File | description |
|---|---|
| dos | Total DOS calculated reading the eigenvalues from report.txt file. |
| pdos-s | PDOS of s-orbitals calculated reading the project.txt file for a given atom. |
| pdos-p | PDOS of p-orbitals calculated reading the project.txt file for a given atom. |
| pdos-d | PDOS of d-orbitals calculated reading the project.txt file for a given atom. |
| tdos | Total DOS: summation of all PDOS calculated reading the project.txt file. |
| pdos-spd | Summation pdos-s + pdos-p + pdos-d for an specified atom calculated reading the project.txt file. |

### 3.2.3 Format

The formats of all output files are the same, containing the DOS or PDOS (in arbitrary units) versus energy in eV. Here is a sample output:

dos/tdos

```
  -12.990973796     0.000015581
  -12.964159962     0.000045690
  -12.937346128     0.000125686
  -12.910532293     0.000324336
  -12.883718459     0.000785140
  -12.856904458     0.001782977
  -12.830090623     0.003798278
  -12.803276622     0.007590564
  -12.776462788     0.014229965
  -12.749648953     0.025025232
     :
     :
```

pdos-s

```
-12.4278824   0.0420946
-12.4010684   0.0329118
-12.3742546   0.0250939
-12.3474406   0.0198658
-12.3206268   0.0182271
-12.2938131   0.0210231
-12.2669991   0.0289676
-12.2401853   0.0425210
-12.2133713   0.0616053
-12.1865574   0.0852457
   :
   :
```

pdos-p

```
-12.6960213   0.0003677   0.0001226   0.0001226   0.0001226
-12.6692073   0.0005339   0.0001780   0.0001780   0.0001780
-12.6423934   0.0007271   0.0002424   0.0002424   0.0002424
-12.6155796   0.0009290   0.0003097   0.0003097   0.0003097
-12.5887656   0.0011134   0.0003711   0.0003711   0.0003711
-12.5619518   0.0012520   0.0004173   0.0004173   0.0004173
-12.5351378   0.0013209   0.0004403   0.0004403   0.0004403
-12.5083241   0.0013082   0.0004361   0.0004361   0.0004361
-12.4815101   0.0012178   0.0004059   0.0004059   0.0004059
-12.4546963   0.0010699   0.0003566   0.0003566   0.0003566
   :
   :
```

pdos-d

```
-12.8300906   0.0000034   0.0000000   0.0000011   0.0000011   0.0000000   0.0000011
-12.8032766   0.0000067   0.0000000   0.0000022   0.0000022   0.0000000   0.0000022
-12.7764628   0.0000126   0.0000000   0.0000042   0.0000042   0.0000000   0.0000042
-12.7496490   0.0000221   0.0000000   0.0000074   0.0000074   0.0000000   0.0000074
-12.7228351   0.0000365   0.0000000   0.0000122   0.0000122   0.0000000   0.0000122
-12.6960213   0.0000565   0.0000000   0.0000188   0.0000188   0.0000000   0.0000188
-12.6692073   0.0000821   0.0000000   0.0000274   0.0000274   0.0000000   0.0000274
-12.6423934   0.0001118   0.0000000   0.0000373   0.0000373   0.0000000   0.0000373
-12.6155796   0.0001429   0.0000000   0.0000476   0.0000476   0.0000000   0.0000476
-12.5887656   0.0001712   0.0000000   0.0000571   0.0000571   0.0000000   0.0000571
   :
   :
```

| Field | Parameter | Kind | description |
|---|---|---|---|
| 1 | *energy* | real | Energy in eV. |
| 2–... | *dos* | real | DOS in arbitrary units. dos: 2nd column, total DOS from report.txt<br>tdos: 2nd column: sum of the *spd* components<br>pdos-s: 2nd column, $s$ component<br>pdos-p: 2nd column, sum of the $p$ components; 3rd-5th columns, $p_x$, $p_y$ and $p_z$ components<br>pdos-d: 2nd-column, sum of the $d$ components; 3rd–7th columns, $d_{z^2}$, $d_{yz}$, $d_{xz}$, $d_{x^2-y^2}$, $d_{xy}$ components |

## 3.3 Examples

### 3.3.1 DOS and PDOS for GaAs bulk

Below we show the total DOS and PDOS calculated for GaAs bulk. The results are shown in Fig. 3.1. Here the Monkhorst-Pack **k**-points mesh was taken as $14 \times 14 \times 14$ (280 **k**-points in the irreducible part of the Brillouin zone) with the initial **k**-point $(0.5, 0.5, 0.5)$ and an energy cutoff of 10 Ry was used. Please note that a large number of **k**-points is needed to obtain a good DOS. The broadening factor was 0.3 eV.

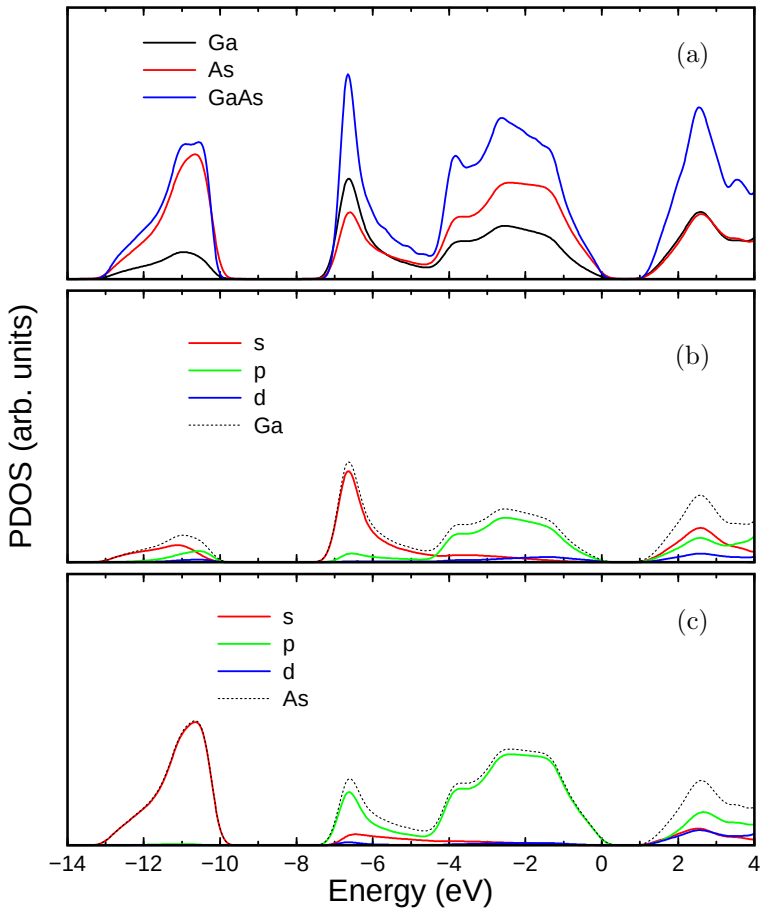



Figure 3.1: DOS and PDOS for GaAs bulk. In (a) the total DOS and PDOS for Ga and As atoms is shown. In (b) the PDOS is decomposed in $s$, $p$ and $d$ components for Ga and (c) for As. The sum `pdos-s`+`pdos-p`+`pdos` is also shown in (b) for Ga and (c) for As.

# Bibliography


[1] M. Bockstedte, A. Kley, J. Neugebauer, and M. Scheffler, Comput. Phys. Commun. **107**, 187 (1997).
URL: <www.fhi-berlin.mpg.de/th/publications/CPC-107-187-1997.pdf>

[2] P. Kratzer, C. G. Morgan, E. Penev, A. L. Rosa, A. Schindlmayr, L. G. Wang, and T. Zywietz, "FHI98MD. Computer code for density-functional theory calculations for poly-atomic systems • User's Manual" (Fritz-Haber-Institut der Max-Planck-Gesellschaft, Berlin, 1999); URL: <www.fhi-berlin.mpg.de/th/fhi98md/doc/FHI98md.pdf>.

[3] L. D. Landau and E. M. Lifshitz, *Elasticity Theory*, Vol. VII of *Course on Theoretical Physics* (Pergamon, 1957), Sec. 4.

[4] F. D. Murnaghan, The compressibility of media under extreme pressures, Proc. Natl. Acad. Sci. USA **30**, 244–247 (1944).

[5] F. F. Janak, V. L. Moruzzi, and A. R. Williams, Phys. Rev. B **12**, 1257 (1975).

[6] G. P. Francis and M. C. Payne, J. Phys.: Condens. Matter **2**, 4395 (1990).

[7] G.-M. Rignanese and Ph. Ghosez and J.-C. Charlier and J.-P. Michenaud and X. Gonze, Phys. Rev. B **52**, 8160 (1995).

[8] J. Tersoff and D. R. Hamann, Phys. Rev. B **31**, 805 (1985).

[9] R. M. Feenstra, Surf. Sci. **299**, 965 (1994).